User Guide of Cargo Data Miner

Xuzhou Qin

10 August 2016

Contents

1 Introduction 3

2 Prerequisites 3

2.1 Directory Structure 4

2.2 Installation on a PC 5

2.3 Deployment on a Shiny Server Pro 6

3 Structure and Features 6

3.1 Cargo IS Database 7

3.1.1 Data loader 7

3.1.2 Top airports/air routes 9

3.1.3 Treemap 11

3.1.4 Yield Evolution 12

3.1.5 Air Waybill Analysis 14

3.1.6 Yield Calculator 14

3.2 Seabury Database 15

3.3 FlightRadar 24 Database 15

3.4 Other features 15

3.4.1 Connection status check 15

3.4.2 Fast selectize input 16

4 Future Development 16

5 Appendix: Troubleshooting 16

5.1 Install Oracle Database instance client 16

5.2 Download and nstall an R package manuelly 17

Abstract

This is the user guide of *Cargo Data Miner*. *Cargo Data Miner* is an R-Shiny Dashboard application developed by Airbus Freighter Marketing department (CSMX). The initiative is to build a platform where everybody could query and analyze cargo data. In this document, detailed information and instruction are provided in order to facilitate the usage and maintenance of the application. For more information, the author could be reached at xuzhou.qin@gmail.com.

Acknowledgement

The present work benefited from Oliver Von-Tronchin, tutor of my internship and director of Freighter Marketing (CSMX), Sebastian Aponte, Nicolas Chauviere-Courcol, colleagues of mine in CSMX, Federico Semeraro and Andy Williams, for their inspiration of the project *LiMA*, Pierre Binet, Didier Gauzy, Aurelien Turina, Sam Crawshay Jones, Lionel Cousseins, Yves Renard and all my colleagues of Airbus who provided valuable comments and ideas to the undertaking of the project here.

1 Introduction

Aviation market is a highly complex and data-dependent market. Inside Airbus, numbers of different servers are deployed to stock, manage and provide database services to support our daily work. It is striving to introduce data digitalization as a core task of its future plan. The raising importance of data has already grabbed the attention of the managers and it will never stop growing. However, face to the massive amount of data, challenge still remains.

At the company level, the main challenge comes from the database management. Too many databases are stocked seperately with different **database management system** (DBMS). The databases are quite isolated from the others. In other words, except people who frequently use one database, others could not even know its existence. Then, the different DBMS cause inconvinience in database management. For example in the freighter marketing department (CSMX), three marketing databases, *Cargo IS*, *Seabury* and *FlightRadar 24* are stocked separately in three data servers using three different DBMS (Oracle Database, MySQL and Microsoft SQL server).

At the department level, the major challenge is the lack of an efficient and user-friendly tool for the data extraction and analysis process. Even thought the traditional approach of database management, SQL, is widely used for data manipulation. But in order to use it efficiently, you must have some knowledge of programming and you have to suffer the difficulty in interfacing. As SQL is not a common tool inside marketing department, interacting with the data by using SQL may not be a suitable choice. As to the data analysis tool, Excel is one of the most widly used software. But it suffers from some innate limitations facing the increasing amount of data.

There are also some Airbus-made tools for marketing department, like *Acper*, *OAG Builder*, *route06*, etc. These are very powerful and sophisticated programs dedicated for specific usage since a very long time. They are irreplaceable for some tasks but they are still not perfect. For example, in terms of learning difficulty, they both have a steep learning curve. Some special training session are needed to master these tools. And from a graphical design's point of view, they often remind me about the "old fashioned" software interface, which may be not enough intuitive and user-friendly. The need to visualize the analytic results in a modern way is thus urgent.

Due to these facts, data digitalization inside Airbus should be done from two level. We will need an uniform and standardized plateform to stock and manage data at the company level. And inside the departments, new data mining and analysis tools are needed to support the works in the big data era. Some moves have already been realised inside Airbus. **BIO** is a platform which provide an automatic aggregation of different databases. **Foundry** is a cloud-based platform for collaboratively big data analysis built by Palantir Technologies.

This document provide a solution for the development of a data analysis tool by using **R Shiny Dashboard**. **Shiny Dashboard** is a framework to create a light weight, powerful and interactive web application. It uses a reactive programming model so that outputs change instantly as users modify inputs. It is esay to use, no web designing language required. But further customization could be easily done with some HTML, CSS and JavaScript knowledge.

In the first section of this document, installation and deployment prerequisites are presented. Then, detailed fonctionalities and the application structure are stated in section two in order to provide technical support for using and maintenance. In the third section, some suggestions for the future development of this application are proposed. The technical appendix could be found in the last section of the document.

2 Prerequisites

Cargo Data Miner (CDM) was developped with **Microsoft R Open 3.2.5**, which is a enhanced distribution of R. It support by defaut multi-thread computing and it is perfectly compatible with the original **R**. The user interface used during the development is **RStudio**, which the the most common IDE of R.

To make **CMD** work properly, there are two main methods.

– Install and run it locally on a PC for personal use.

– Deploy it on a Shiny server so that CDM could be reached by an IP adress.

In this section, I will present both the two methods.

2.1 Directory Structure

The following figure presents the structure of the CDM directory.

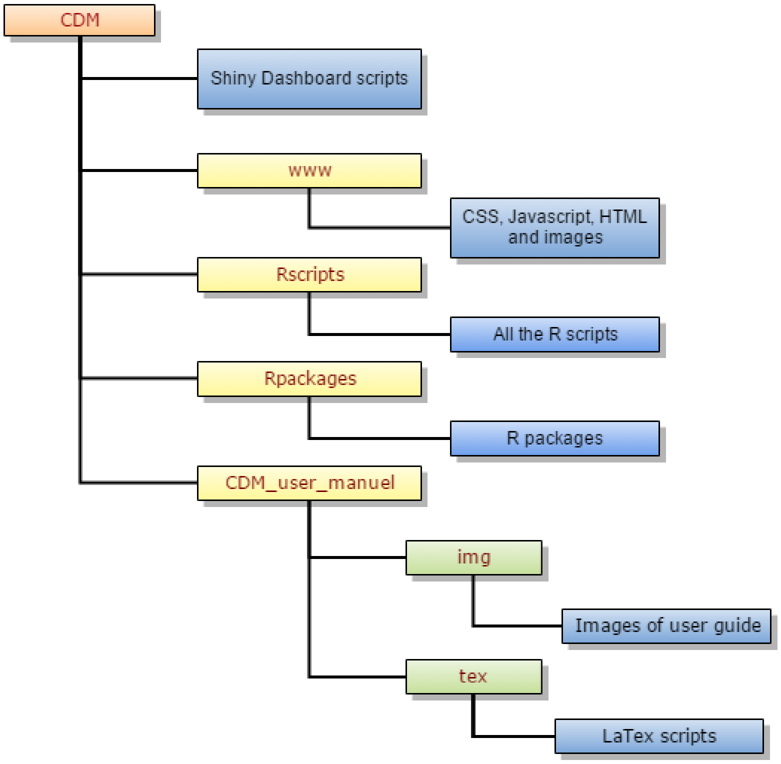


Figure 1: CDM directory structure

The main directory `~/CDM` contains Shiny Dashboard scripts (*ui.R*, *server.R* and *global.R*) and three directories:

- `/www`
- `/Rscripts`
- `/Rpackages`
- `/CDM_user_manuel`

All the images, HTML, CSS and JavaScript used in **CDM** should be placed in `~/CDM/www`. All the R scripts called by the application should be placed in `~/CDM/Rscripts` and all the R packages (for local installation) should be placed in `~/CDM/Rpackages`.These three directories are the essential parts of **CDM**.

In `~/CDM/CDM_user_manuel`, you will find the code of this use guide (written in R markdown) and its resources. All figures appeared in this document should be placed in `~/CDM/CDM_user_manuel/img` and all the LaTex scripts called from the R markdown code will be found in `~/CDM/CDM_user_manuel/tex`.

2.2 Installation on a PC

CDM could run on a personal computer having installed R and a web browser with the correct configuration. Before the first use, some steps have to be followed.

- Step 1: Get R and other required software:
  - Install **Microsoft R Open 3.2.5** from https://mran.revolutionanalytics.com/download/ (to turn on the multi-thread computing support, install the MLK package as well), or you can install the original **R** (64-bit) from https://cran.r-project.org/mirrors.html.
  - Install **RStuido** from https://www.rstudio.com/products/rstudio/. This is optional but highly recommanded.
  - **CDM** works with Internet Explorer but you could also install Google Chrome for the best performance.
  - Install **Oracle Database Instant Client (11.2.0.3) EN_W764** from PC service. Or you can download it from Oracle website (Make sure it is the 64-bit version). If error of Oracle occurs during the execution of **CMD**, see Appendix: Troubleshooting for help.
- Step 2: Connection Setup
  - **CMD** will need the access right of three different databases. They are: **p595dodmp01** (Cargo IS, Seabury), **fr0-bio-p01** (BIO) and **fr0-dmds-p01** (FlightRadar 24). The user name and password are already pre-filled and there is no need to change, except for **BIO** database. Because you will need to grant the access right to your own *Windows* account to make the connection. To do so, you could ask *Aurelien Turina* for further information.
  - `~/CDM/global.R` contains the connection parameters of these databases.

```
##########################################################
#' CONNECTION PARAMETERS
#'
#' Modify it before the first execution of the application
##########################################################
#' ROracle connection string
#' for Cargo IS, Seabury
host = 'p595dodmp01'
port = 1521
sid = 'DBUPA269'
username_cargois <- 'CARGO'
password_cargois <- 'dbu1_cargo'


#' RODBC MS SQL server connection string
#' for BIO
driver_bio <- 'SQL Server'
server_bio <- 'fr0-bio-p01'
port_bio <- 10335
username_bio <- ''
password_bio <- ''
trusted_connection_bio <- TRUE # if TRUE connect with windows login and pw


#' RMySQL MySQL connection string
#' for FlightRadar 24
```

```
user_fr24 <- "user_ext"
password_fr24 <- "fr0-dmds-p01"
dbname_fr24 <- 'FR24'
host_fr24 <- "fr0-dmds-p01"
```

- Step 3: Now you can open RStudio.
  - Before running **CDM**, if you have more than one version of R installed on your computer, make sure that the R version is `[64-bit] ~/MRO_3.2.5` (the name could be varied)
  - Click on the **run** button on the top right of the text editor aera to execute the application. The application will automatically check if all the required R packages are installed and it will install those which are not installed from **CRAN** (Comprehensive R Archive Network). **CRAN** is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. However, due to the internet connection configuration inside Airbus, if any of the required packages fails to download, you will have to download and install it manuelly (See Appendix for detail).

Now you are ready to use **CDM** :)

2.3 Deployment on a Shiny Server Pro

Shiny Server Pro is a server program that makes Shiny application available over the web.

At this time, Shiny Server can be run on Linux servers with explicit support for Ubuntu 12.04 or greater (64 bit) and CentOS/RHEL 5 (64 bit) or greater. Multiple Shiny applications on multiple web pages could be hosted with the same Shiny Server.

It can be downloaded from https://www.rstudio.com/products/shiny/shiny-server/.

For the detail instruction of the configuration and installation of Shiny Server Pro, see **Shiny Server Professional v1.4.2 Administrator's Guide** on http://docs.rstudio.com/shiny-server/.

3 Structure and Features

A Shiny dashboard application could contains three sources scripts:

- a user-interface script: `ui.R`
- a server script: `server.R`
- a global environment script: `global.R`

There is another structure of Shiny application where all the three main scripts could be integrated into one single scipt `app.R`. But here we only talk about the "traditional" structure.

The UI script controls the layout and appearance of the app. The server script contains the instructions that your computer needs to build your app. And the glabal script defines objects in the global scope.

Reactivie programming is the fundamental of a Shiny Dashboard application. The main idea is to make it easy to wire up *input values* from a web page, trigger the R code to be (re)executed, and then have the results of your R code be written as *output values* back to the web page.

The following figure shows the three elements in Shiny reactive programming. **Reactive source** typically is the user's input through a web browser. It could trigger the execution of an R code. The output of the code is represented by the **reactive endpoint**. **Reactive conductor** could be placed between the sources

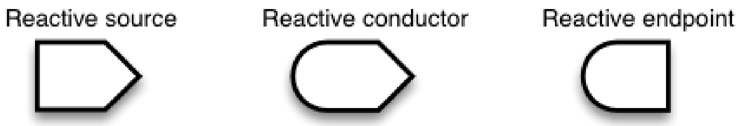
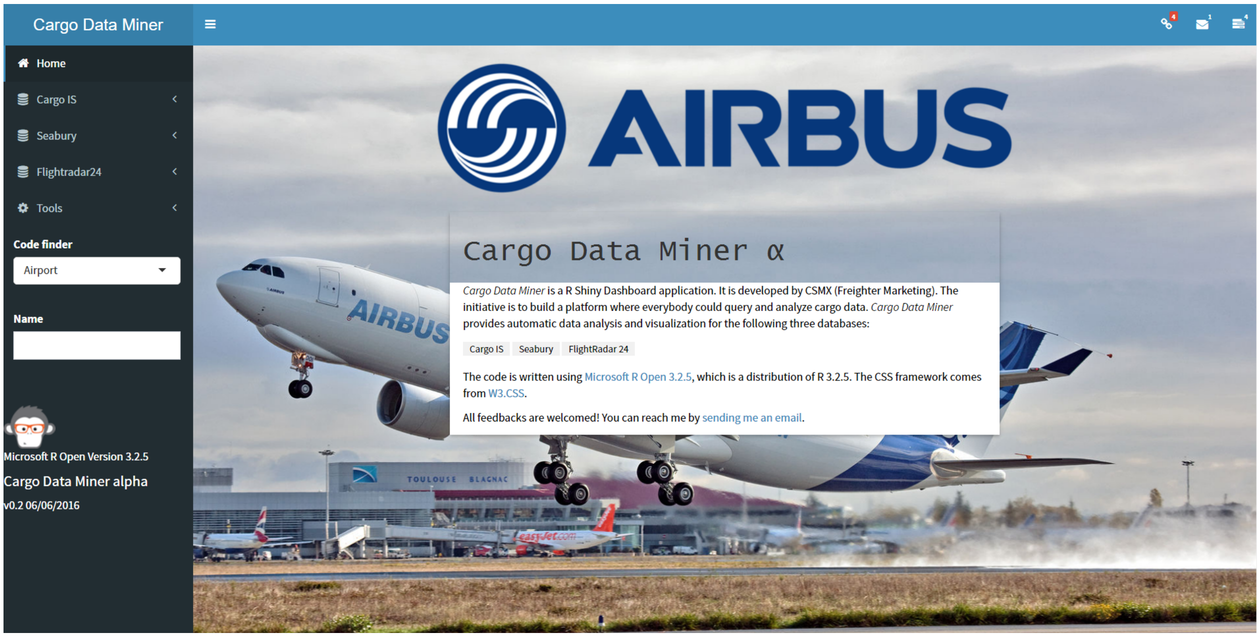


Figure 2: Essential objects in reactive programming

and endpoints. Generally they are used to encapsulate the computationally expensive operations in order to augment the code performance.

After launching **CDM**, you will see the following welcome page. **CDM** has three parts: a header, a sidebar and a body. The header show the application name and some notification[1], for example the database connection status, the app version and the developing progres. The sidebar could help you to navigate between different tabs The functionalities of these widgets will be presented in the following sections. Finally the body is the main aera where you will be able to interact with **CMD** and see the output.



Cargo Data Miner α

Cargo Data Miner is a R Shiny Dashboard application. It is developed by CSMX (Freighter Marketing). The initiative is to build a platform where everybody could query and analyze cargo data. *Cargo Data Miner* provides automatic data analysis and visualization for the following three databases:

Cargo IS Seabury FlightRadar 24

The code is written using Microsoft R Open 3.2.5, which is a distribution of R 3.2.5. The CSS framework comes from W3.CSS.

All feedbacks are welcomed! You can reach me by sending me an email.

Figure 3: **CMD** home page

Five different tabs are displayed on the sidebar. Each tab contains also several subtabs.

3.1 Cargo IS Database

Cargo IS database is the first major conponent of **CDM**. It provides airport level Air Waybill data, including airfreight charges, weight, numbers of shipmens and it is the only ressource of cargo yield information for Airbus.

3.1.1 Data loader

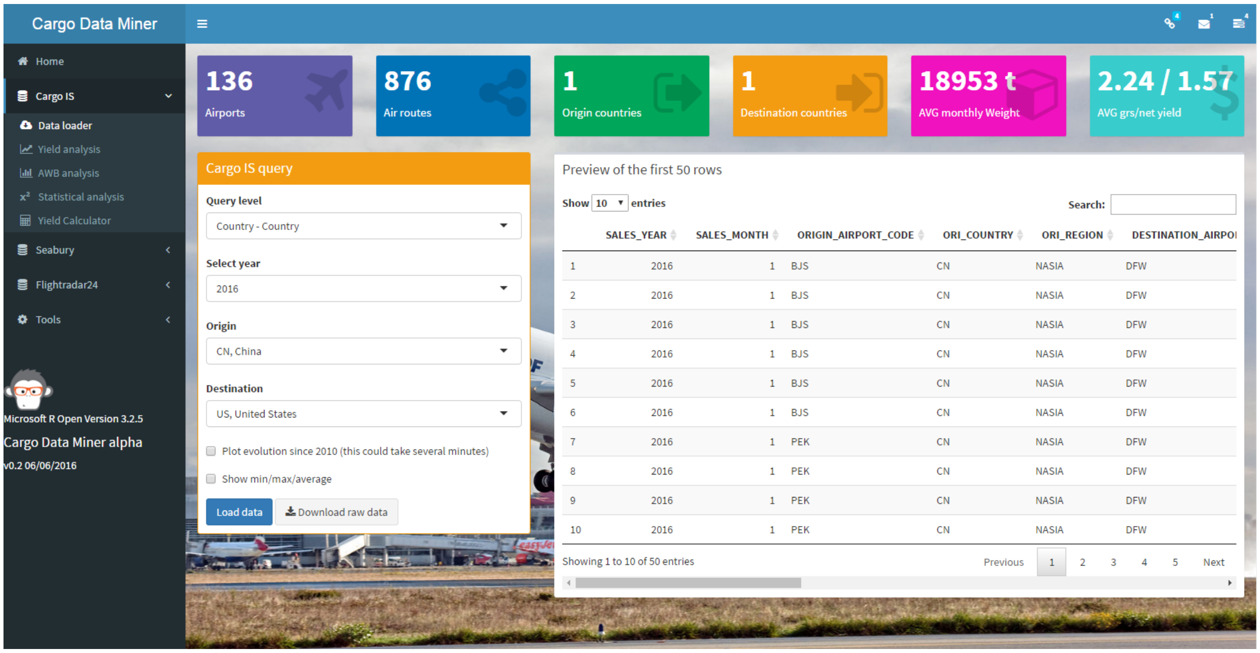
3.1.1.1 Instruction

[1] Some features are still under development.

The first step is about how to load Cargo IS data. In the first dropdown menu, we could select the level of data granularity to be queried from Cargo IS (*e.g.*, airport to airport, country to country, region to region, etc). Then the year and the **O&D** (**O**rigin and **D**estination) of airfreight should be precised. You can check the box *"Plot evolution since 2010"* to load data of all the available year (2010-2016). When the box is checked, the year input will be ignored.

After selecting the right parameters, click on **Load Data** buttom. Some summerize of the queried dataset will be shown on the top of this tab. A preview of the 50 first rows will appear on the right. You could do a quick check of the data. The data table could be downloaded by simply clicking on the **Download Data** button.

Now the data table is loaded and ready to be used!



136 Airports

876 Air routes

1 Origin countries

1 Destination countries

18953 t AVG monthly Weight

2.24 / 1.57 AVG grs/net yield

Cargo IS query

Query level: Country - Country

Select year: 2016

Origin: CN, China

Destination: US, United States

□ Plot evolution since 2010 (this could take several minutes)

□ Show min/max/average

Preview of the first 50 rows

Show 10 entries

Search:

	SALES_YEAR	SALES_MONTH	ORIGIN_AIRPORT_CODE	ORI_COUNTRY	ORI_REGION	DESTINATION_AIRPO
1	2016	1	BJS	CN	NASIA	DFW
2	2016	1	BJS	CN	NASIA	DFW
3	2016	1	BJS	CN	NASIA	DFW
4	2016	1	BJS	CN	NASIA	DFW
5	2016	1	BJS	CN	NASIA	DFW
6	2016	1	BJS	CN	NASIA	DFW
7	2016	1	PEK	CN	NASIA	DFW
8	2016	1	PEK	CN	NASIA	DFW
9	2016	1	PEK	CN	NASIA	DFW
10	2016	1	PEK	CN	NASIA	DFW

Showing 1 to 10 of 50 entries

Figure 4: An example query of the 2015 airfreight from *China* to *the USA*.

3.1.1.2 Explaination

The data loading procedure will call the `CargoIS.SQL.query` function. The function will take the user parameters as inpus, generate an appropriate `Oracle SQL` code and send the query to the data server. The returned dataset is thus stocked in the global scope and it is ready to be used by other functions.

```r
# EXAMPLE
# Query airfreight data of 2015 with origin country = China, destination = USA
dat <- CargoIS.SQL.query(CargoDB, level = 'C2C',
                         ORG = 'CN',
                         DST = 'US',
                         year = 2015)
```

The six value boxes on the top of the page are created by the function `valueBox(value, subtitle, icon, color, width, href)` based on the output dataset. They represent:

- **Airports**: Numbers of unique airports appeared in the dataset.
- **Air routes**: Numbers of unique airport pairs (Origin + Destination) in the dataset.

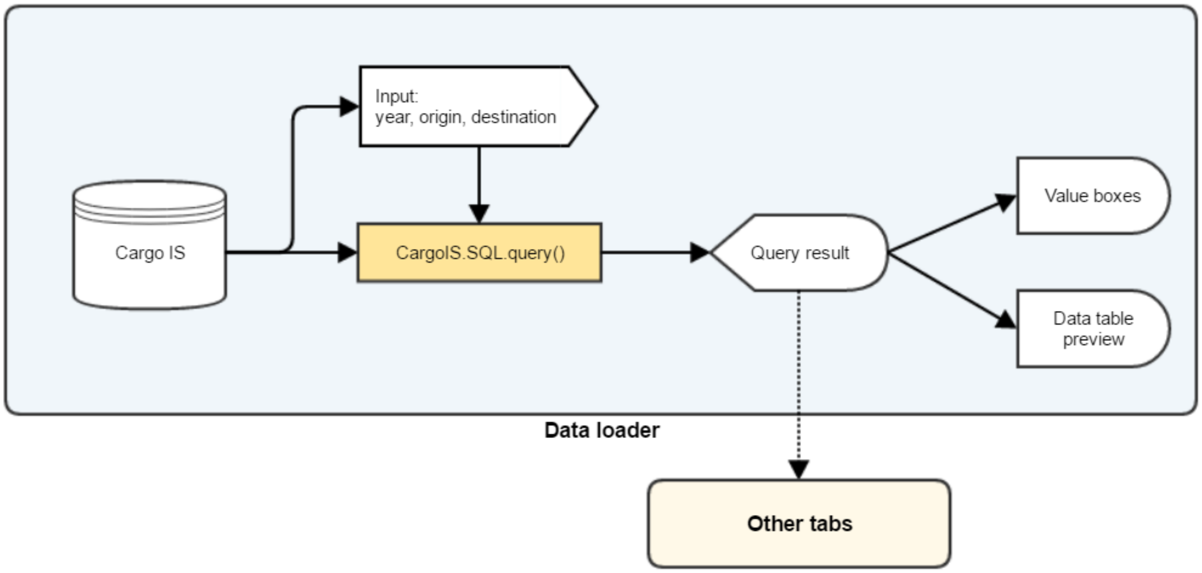


Figure 5: Flowchart of data loading process

- **Origin country**: Numbers of unique origin country(ies).
- **Destination country**: Numbers of unique destination country(ies).
- **AVG monthly weight**: Arithmetic mean of weight.

$$Average\ monthly\ weight = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} weight_{ij}}{n}$$

, where i is the number of months, j is the weight break.

- **AVG gross/net yield**:

$$Average\ gross\ yield = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} charges_{ij} + surcharges_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} weight_{ij}}$$

$$Average\ net\ yield = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} charges_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} weight_{ij}}$$

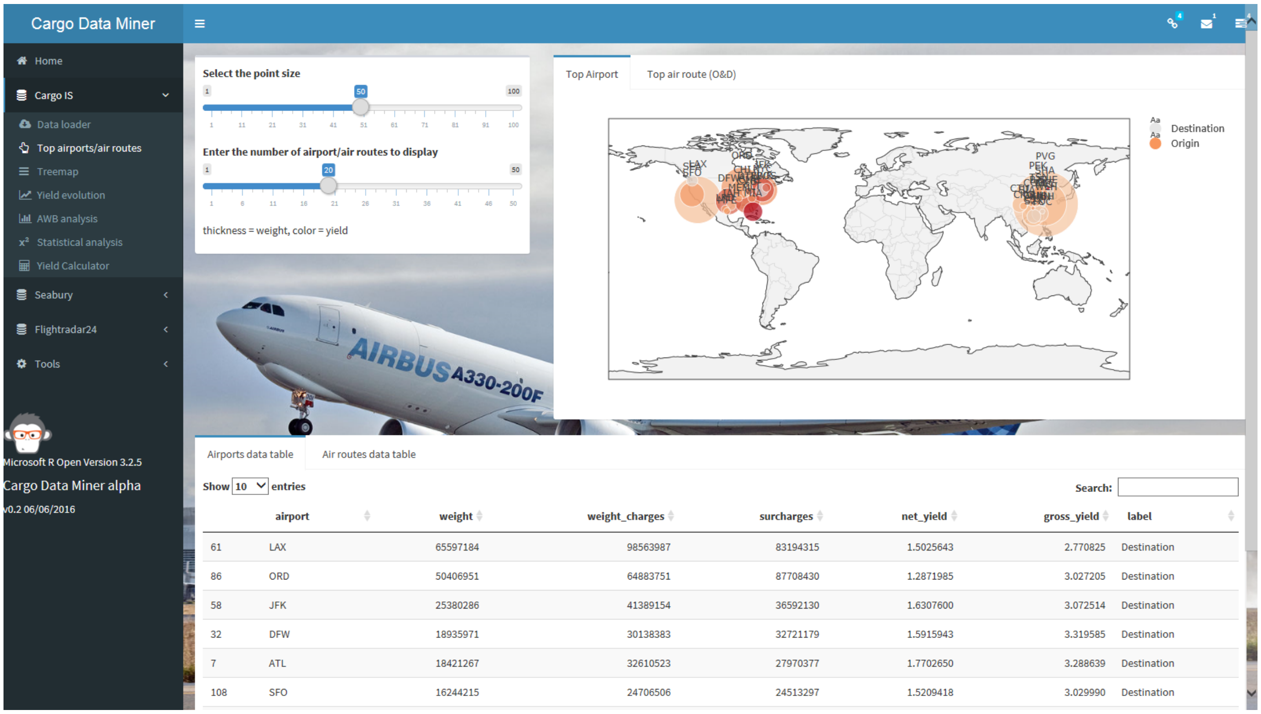
, *charges* are fees directly related to the weight of the shipment and *surcharges* contain all the other fees (fuel surcharges, security surcharges, taxes, etc.)

3.1.2 Top airports/air routes

When the data loading procedure is complete, you can go to the following tabs to see some visualization output. After clicking on the “Yield Analysis” tab, **CDM** will automatically proceed and visualize the loaded dataset.

3.1.2.1 Instruction

The first part of tab contains visualization of the **top *n* airports and air routes**. Figure 5 shows in the map the top 20 origin and destination airports. The surface of the circle represents the weight of shipment going through this airport. The darkness of the color represents the yield of the airport. The charts are



Select the point size: 50

Enter the number of airport/air routes to display: 20

thickness = weight, color = yield

Airports data table | Air routes data table

Show 10 entries

Search:

	airport	weight	weight_charges	surcharges	net_yield	gross_yield	label
61	LAX	65597184	98563987	83194315	1.5025643	2.770825	Destination
86	ORD	50406951	64883751	87708430	1.2871985	3.027205	Destination
58	JFK	25380286	41389154	36592130	1.6307600	3.072514	Destination
32	DFW	18935971	30138383	32721179	1.5915943	3.319585	Destination
7	ATL	18421267	32610523	27970377	1.7702650	3.288639	Destination
108	SFO	16244215	24706506	24513297	1.5209418	3.029990	Destination

Figure 6: Yield Analysis tab, top 20 airports of CN-US

interactive, you can zoom in/out with the mouse scroll wheel. You can also click on the legend to display/hide the origin/destination airports.

The value of n could be modified by dragging the slider bar. You could also modify the size of the circle if sometimes the defaut size is too big/small.

The **Top air routes (O&D)** tab of the visualization chart shows the weight and yield of the most important air routes. Origin and destination airports are linked by blue curves with different thickness and darkness. The **thicker** the curve, the **more** the total weight and the **darker** the blue, the **higher** the yield.

The last two tabs (Airports data table and Air routes data table)show the two data tables that **CDM** uses to create the map. It could be sorted by variables and it could be directly copy-pasted into an Excel sheet.

3.1.2.2 Explaination

An R package `Plot_ly` was used to create the visualization map. From `/Scripts/Plot.airport.html.R`, an user defined function `Plot.airport.html()` is called.

```r
#' [Plot.airport.html] could be used to visualize the top n airports/air routes
#' When the parameter table = TRUE, it will output the datatable that it will
#' use for the data visualization.
#'
#' @param dataset: data frame with five columns
#' c('ORIGIN_AIRPORT_CODE', 'DESTINATION_AIRPORT_CODE', 'WEIGHT_CURRENT_YEAR',
#' 'WEIGHT_CHARGES_CURR_YEAR_USD', 'OTHER_CHARGES_CURR_YEAR_USD')
#'
#' @param projection: Mercator or orthographic
#'
#' @param top: numbers of airport/air routes to be displayed
#'
```

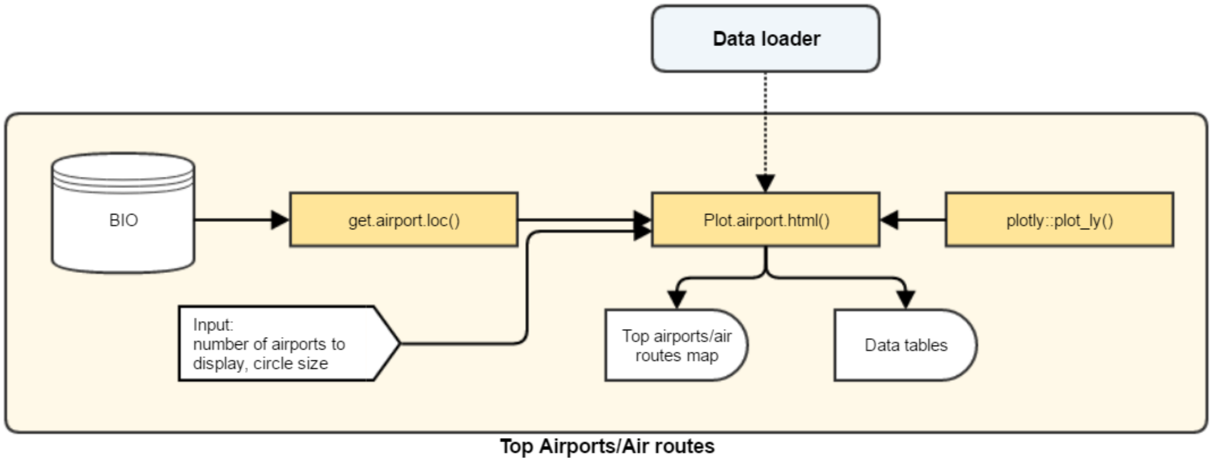


Figure 7: Flowchart of top airports/air routes tab

```
#' @param plot.airport: if TRUE, plot airport, if FALSE, plot air routes
#'
#' @param pt.size: int, point size
#'
#' @param table: if TRUE, output datatable, otherwise, output map

 Plot.airport.html(dataset, projection = 'Mercator', top = 20, plot.airport = TRUE,
                   pt.size = 800000, table = FALSE)
```

`Plot.airport.html()` will call `plotly::plot_ly()`. `plotly::plot_ly()` provides a simple way to add scatters/lines/curves to a map. It will only need the coordinates of the O&D airports, which we could easily get from BIO database by doing

```
source('/Scripts/get.airport.loc.R')
get.airport.loc()
```

Figure 7 shows how the function `plot.airport.html()` works.

3.1.3 Treemap

3.1.3.1 Instruction

The tab **Treemap** visualizes the relationship between different **nodes** (airports, countries and regions). Each node is displayed as a rectangle, sized by the total airfreight weight of this node and colored by the average gross yield of the node.

You can click on the rectangle to see its child nodes[2] (if exist), and right-click to see its parent node. A scale is shown on the top right to indicate the yield level. The tab **Table** will render the used data table for the visualization. The table contains four columns, *Name*, *Parent*, *Weight*, *GrossYield*.

Currently there is no fitable solution to generate a static downloadable image. So you may take a screen shot if you want to add the chart to your slides.

[2] For example, **country** is the child node of **region**, and it is the parent node of **airport**.

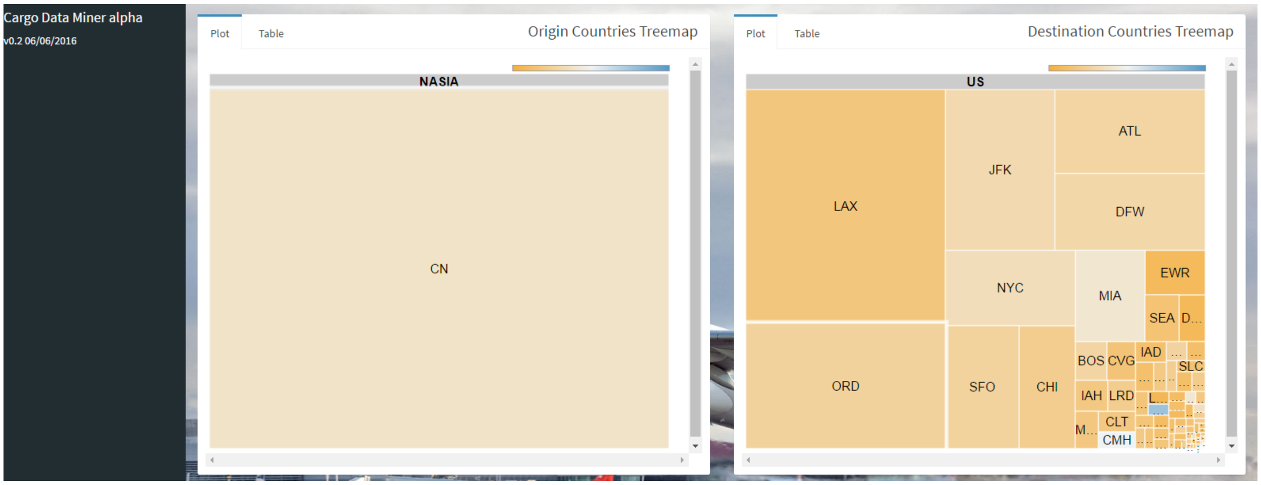


Figure 8: Treemap of airports for CN and US

3.1.3.2 Explaination

To create an interactive treemap, I created an user-defined function `treemap_cargois_gvis()`, which will call `googleVis::gvisTreeMap()`. The package `googleVis` is the R interface to **Google Charts API**. It generates an HTML code which could be integrated into R Shiny application. It could be further customized by using Javascripts.

```
# Example:
treemap_cargois_gvis(data, ORG = FALSE, PLOT = TRUE)

#' data is a data.frame object generated by Cargois.SQL.query()
#' If ORG = TRUE, then create treemap for the origin airports,
#' else for the the destination airports
#' If PLOT = FALSE, then the output will be a data.frame object
#' (instead of a plot)
```

The function `treemap_cargois_gvis()` will convert the queried Cargo IS data table into a four-column dataset (*Name*, *Parent*, *Weight*, *GrossYield*), which could be used by `googleVis::gvisTreeMap()`.

```
# Example of gvisTreemap
gvisTreeMap(data = datatable, idvar = 'Name', parentvar = 'Parent', sizevar = 'Weight',
                colorvar = 'GrossYield',
                options=list(fontSize=16,
                             minColor='#F5B041',
                             midColor='#F0F3F4',
                             maxColor='#5499C7',
                             headerHeight=20,
                             fontColor='black',
                             showScale=TRUE)
                            # options, see
                            # https://developers.google.com/chart/interactive/docs/
            )
```

3.1.4 Yield Evolution

Yield is the most important information provided by Cargo IS database.

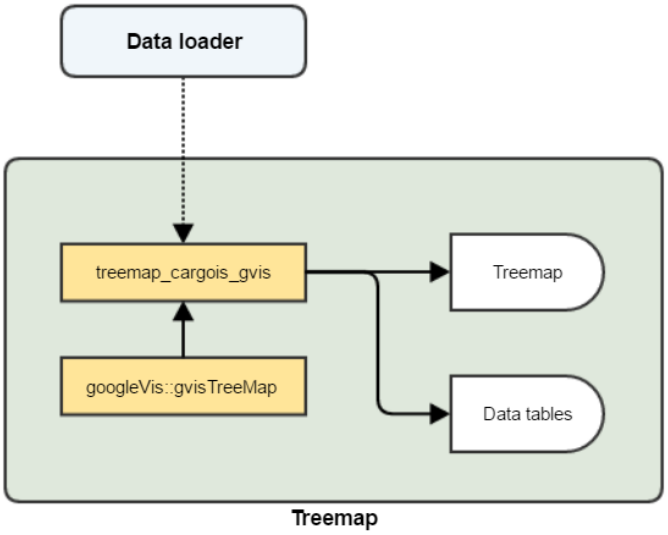


Figure 9: Flowchart of Treemap tab

3.1.4.1 Instruction

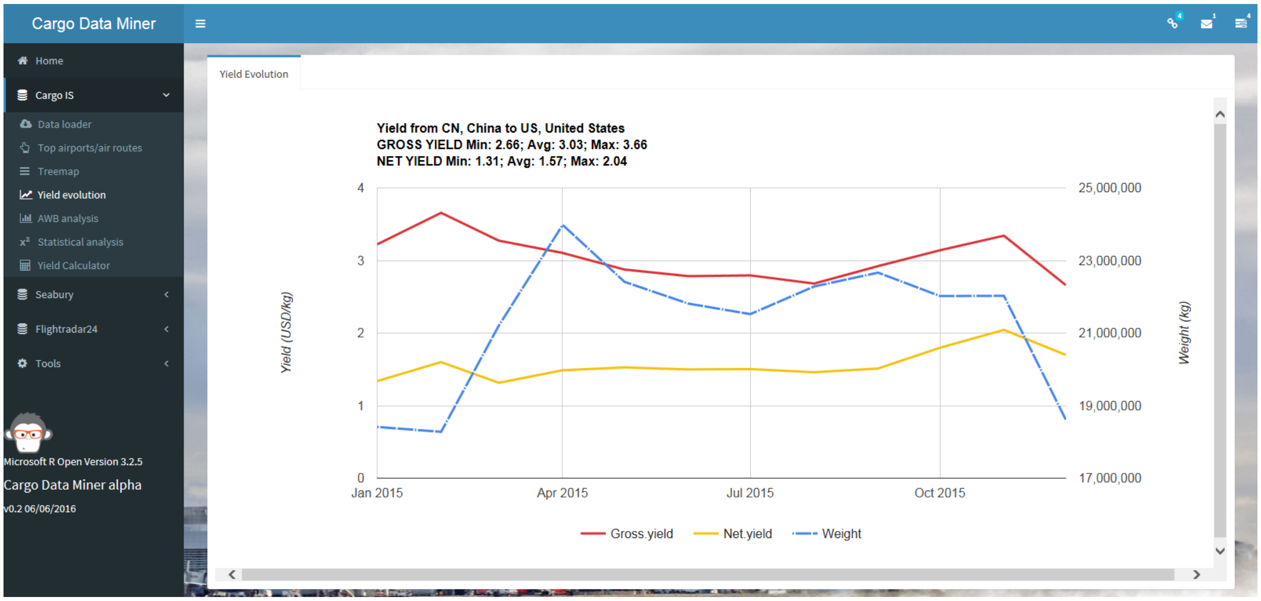


Figure 10: Airfreight weight and yield evolution for CN and US

The next tab **Yield evolution** generates an airfreight evolution plot. **CDM** calculates and render a line chart showing the evolution of gross yield, net yield and airfreight weight during the selected period of time. The minimum, maximum and average value of the output are shown in the plot's subtitle.

The x-axis represents the available months during the selected period. If the box "Plot evolution since 2010" in the data loaded page is checked, all the available data will be visualized (but the computing will also take a longer time).

You could click on the curve to see detailed information of the choosen data point.

A data table appears on the bottom of the page. It contains all the data of the line chart. You could click on the download button to save it to the local.

3.1.4.2 Explaination

I would like to integrate the three curves (net yield, gross yield and weight) into one sigle chart. Thus two different y-axis are needed. The well known package `ggplot2` does not support two different y-axis in one chart so I will still use `googleVis` to generate the line chart.

```
# Example of givsLineChart
# with gvisLineChart, we could easily create a line chart with 2 y-axis

gvisLineChart(data, xvar = "xaxis", yvar = c('yaxis_1', 'yaxis_2', 'yaxis_3'),
              options = list(series="[
                                {targetAxisIndex: 0}, # set the target y-axis
                                {targetAxisIndex: 0},
                                {targetAxisIndex: 1}
                                ]"
                              )
              )
```

3.1.5 Air Waybill Analysis

An air waybill is a receipt issued by an international airline for goods and an evidence of the contract of carriage. It is the most important document issued by the carrier (directly of throught an authorised agent). Cargo IS contains the records the weight data and it regroups the air waybills into six categories, which are called **weight breaks**, regarding to its weight.

They are:

- 0 to 50 kg
- 50 to 100 kg
- 100 to 300 kg
- 300 to 500 kg
- 500 to 1000 kg
- larger than 1000 kg

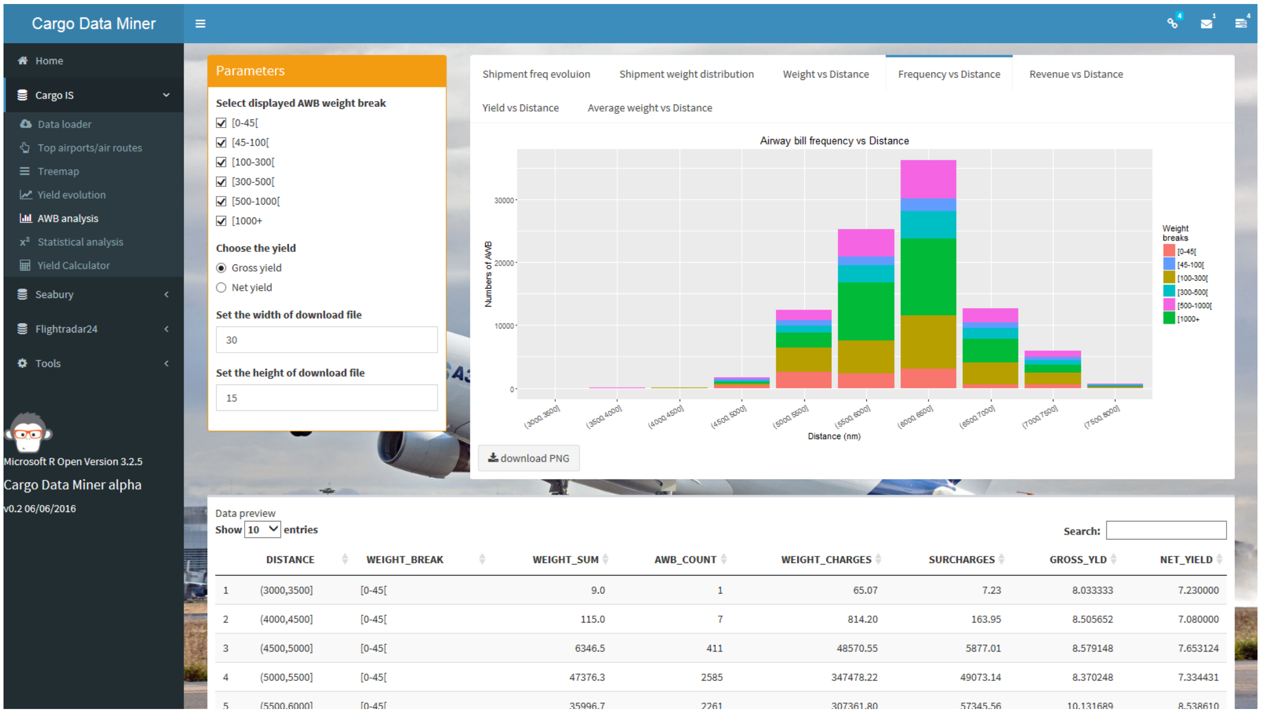
3.1.5.1 Instruction

3.1.5.2 Explaination

3.1.6 Yield Calculator

The last subtab of Cargo IS

3.1.6.1 Instruction



Select displayed AWB weight break

☑ [0-45[
☑ [45-100[
☑ [100-300[
☑ [300-500[
☑ [500-1000[
☑ [1000+

Choose the yield

● Gross yield
○ Net yield

Set the width of download file: 30

Set the height of download file: 15

Data preview

Show 10 entries

Search:

	DISTANCE	WEIGHT_BREAK	WEIGHT_SUM	AWB_COUNT	WEIGHT_CHARGES	SURCHARGES	GROSS_YLD	NET_YIELD
1	(3000,3500]	[0-45[	9.0	1	65.07	7.23	8.033333	7.230000
2	(4000,4500]	[0-45[	115.0	7	814.20	163.95	8.505652	7.080000
3	(4500,5000]	[0-45[	6346.5	411	48570.55	5877.01	8.579148	7.653124
4	(5000,5500]	[0-45[	47376.3	2585	347478.22	49073.14	8.370248	7.334431
5	(5500,6000]	[0-45[	35996.7	2261	307361.80	57345.56	10.131689	8.538610

Figure 11: Air waybill analysis

3.2 Seabury Database

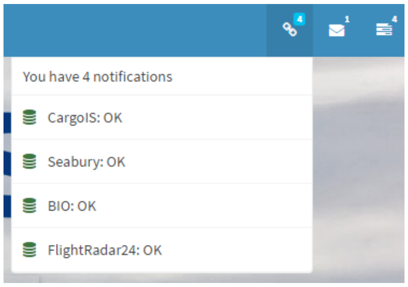
3.3 FlightRadar 24 Database

3.4 Other features

3.4.1 Connection status check

The first link icon on the top right shows the conncection status of **CDM** to the databases. If all the databases are seccesfully connected, the color around the number will be blue. Otherwise the backgroud color will turn red.

You can click on the icon to see the detailed connection information.



You have 4 notifications

CargoIS: OK

Seabury: OK

BIO: OK

FlightRadar24: OK

Figure 12: All the connections are succeed

3.4.1.1 Explaination

Each time when a new session of **CDM** starts, it will try to make a connection to these databases and check the class of the connector. If the class of all the connectors are not `'try-error'`, the value `OK` will be given to the variables `statu_cargois`, `statu_bio` and `statu_fr24`.

3.4.2 Fast selectize input

4 Future Development

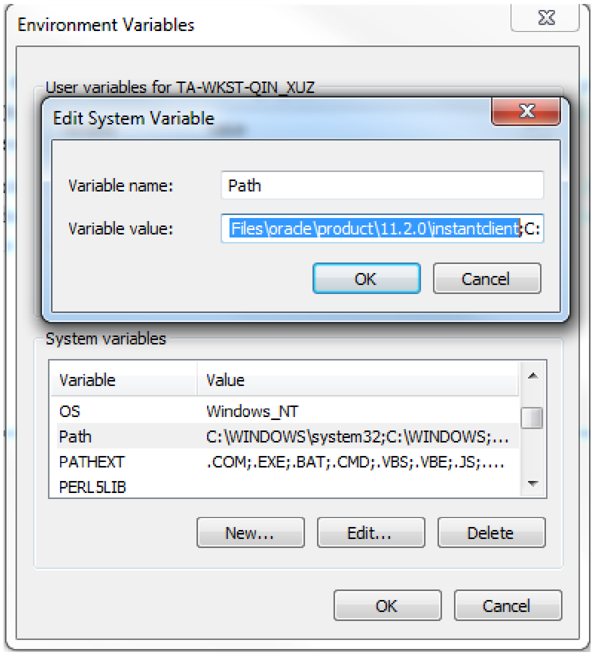
5 Appendix: Troubleshooting

5.1 Install Oracle Database instance client

Oracle Database Instant Client (11.2.0.3) EN_W764 could be found in **PC Service**, or it could be downloaded from http://www.oracle.com/technetwork/topics/winx64soft-089540.html (you may need to create an account to be able to download it). After downloading the client, extract it to a folder that you could find again.

If **CDM** could not be executed due to errors of Oracle instant client, you can check if the environment variables of Windows is set correctly by doing the following procedure.

- Open the properties window of the computer.
- Click on "**Advanced system settings**"
- Select the tab "**Advanced**" of the appearing dialog box.
- Click "**Environment Variables**" button on the bottom.
- Find the variable "**Path**", make sure that the path of your Oracle instance client is added. If not, add it.



Variable name: Path

Variable value: Files\oracle\product\11.2.0\instantclient;C:

Variable	Value
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PERL5LIB	

Figure 13: Check the "Path" system variable

If the problem still exists, check the **ROracle** package installation instruction on http://cran.us.r-project.org/web/packages/ROracle/INSTALL

5.2 Download and nstall an R package manuelly

Some of the R package could not be installed directly from **CRAN**, for example the package `ROracle`. In this situation, the manuel installation will be needed.

Here I will show the installation process for `ROracle`.

- Firstly, download `ROracle_1.2-1.zip` from *oracle.com* (http://www.oracle.com/technetwork/database/database-technologies/r/roracle/downloads/index.html)
- Secondly, place the .zip file into `~/CDM/Rpackages`
- Then, instead of running

```
install.packages('ROracle')
```

run

```
install.packages('Rpackages/ROracle_1.2-1.zip', repos = NULL, type="source")
```

After the installation, the package could be called by

```
library(ROracle)
```